

Amendments to the Claims

Please amend Claims 12, 14-15, 22, 24, 31, 34, and add new Claim 35, all as shown below. Applicant respectfully reserves the right to prosecute any originally presented or canceled claims in a continuing or future application.

1.-11. (Canceled).

12. (Currently Amended) A system for ~~interleaving~~ supporting resource enlistment synchronization, comprising:

an application server with a plurality of threads, running on one or more processors;

~~a plurality of resource objects, wherein each resource object is wrapped with a wrapper object in a collection of wrapper objects;~~

a transaction manager that manages a plurality of transactions, wherein each transaction is associated with at least one said thread, and the transaction manager operates to be associated with each of the plurality of threads;

a plurality of resource objects, wherein each resource object is wrapped with a wrapper object of a plurality of wrapper objects, wherein the transaction manager maintains and communicates with the ~~collection~~ plurality of wrapper objects to manage resource object enlistment requests from different said threads associated with different transactions;

wherein, ~~after the transaction manager receives~~ upon receiving a request from a thread of the plurality of threads to enlist a resource object of the plurality of resource objects in a transaction, the transaction manager

first checks with a wrapper object of the resource object to see if there is a lock being held on the resource object by another said thread in another said transaction,

if there is a lock, then allows the thread to wait and signal the thread once the lock is freed by the another said thread in the another said transaction,

if there is no lock, then grants a lock to ~~an accessor associated with~~ the thread and holds the lock until an owner of the thread delists the resource object, wherein the wrapper object is used to access the resource object for the thread.

13. (Canceled).

14. (Currently Amended) The system of Claim 12, wherein:
the ~~collection~~ plurality of wrapper objects ~~[[is]]~~ are periodically processed to remove objects that are unused or no longer active.
15. (Currently Amended) The system of Claim 12, wherein:
each of the ~~one or more~~ plurality of resource objects resides in a server node.
16. (Canceled).
17. (Previously Presented) The system of Claim 12, wherein:
the transaction manager uses a priority method to determine which thread will be granted a lock.
18. (Previously Presented) The system of Claim 12, wherein:
after the thread obtains a lock, the thread uses the wrapper object to initiate work on the resource object.
19. (Previously Presented) The system of Claim 12, wherein:
the wrapper object receives a delist call from the transaction manager and sends an end call to the resource object to end work performed by the resource object associated with the thread and release the lock on the resource object.
20. (Previously Presented) The system of Claim 12, wherein:
once the transaction manager enlists the resource object and obtains a lock to the resource object, any attempted enlist from a second thread is blocked.
21. (Canceled).
22. (Currently Amended) A method for ~~interleaving~~ supporting resource enlistment synchronization, comprising:
providing an application server with a plurality of threads, running on one or more processors;

~~wrapping each resource object of a plurality of resource objects with a wrapper object in a collection of wrapper objects;~~

managing, via a transaction manager, a plurality of transactions, wherein each transaction is associated with at least one said thread, and the transaction manager operates to be associated with each of the plurality of threads ~~wherein the transaction manager maintains the collection of wrapper objects to manage resource object enlistment requests from different said threads associated with different transactions;~~

wrapping each resource object of a plurality of resource objects with a wrapper object in a collection of wrapper objects, wherein the transaction manager maintains and communicates with the plurality of wrapper objects to manage resource object enlistment requests from different said threads associated with different transactions;

receiving a request from a thread of the plurality of threads to enlist a resource object of the plurality of resource objects in a transaction at the transaction manager;

first checking with a wrapper object of the resource object, via the transaction manager, to see if there is a lock being held on the resource object by another said thread in another said transaction;

if yes, allowing, via the transaction manager, the thread to wait and signaling the thread once the lock is freed by the another said thread in the another said transaction;

if not, granting, via the transaction manager, a lock to ~~an accessor associated with the thread and holding the lock until an owner of the thread delists the resource object,~~ wherein the wrapper object is used to access the resource object for the thread.

23. (Canceled).

24. (Currently Amended) The method of Claim 22, wherein:

the ~~collection~~ plurality of wrapper objects is periodically processed to remove objects that are unused or no longer active.

25. (Canceled).

26. (Previously Presented) The method of Claim 22, further comprising:

using a priority method to determine which thread will be granted a lock.

27. (Previously Presented) The method of Claim 22, further comprising:
after obtaining a lock, using, via the thread, the wrapper object to initiate work on the resource object.
28. (Previously Presented) The method of Claim 22, further comprising:
receiving, at the wrapper object, a delist call from the transaction manager and sending an end call to the resource object to end work performed by the resource object associated with the thread and release the lock on the resource object.
29. (Previously Presented) The method of Claim 22, further comprising:
once the transaction manager enlists the resource object and obtains a lock to the resource object, blocking any attempted enlist from a second thread.
30. (Canceled).
31. (Currently Amended) A computer-readable storage medium, storing instructions for ~~interleaving~~ supporting resource enlistment synchronization, the instructions comprising the steps of:
providing an application server with a plurality of threads, running on one or more processors;
~~wrapping each resource object of a plurality of resource objects with a wrapper object in a collection of wrapper objects;~~
managing, via a transaction manager, a plurality of transactions, wherein each transaction is associated with at least one said thread, and the transaction manager operates to be associated with each of the plurality of threads wherein the transaction manager maintains the collection of wrapper objects to manage resource object enlistment requests from different said threads associated with different transactions;
wrapping each resource object of a plurality of resource objects with a wrapper object in a collection of wrapper objects, wherein the transaction manager maintains and communicates with the plurality of wrapper objects to manage resource object enlistment requests from different said threads associated with different transactions;
receiving a request from a thread of the plurality of threads to enlist a resource object of the plurality of resource objects in a transaction at the transaction manager;

first checking with a wrapper object of the resource object, via the transaction manager, to see if there is a lock being held on the resource object by another said thread in another said transaction;

if yes, allowing, via the transaction manager, the thread to wait and signaling the thread once the lock is freed by the another said thread in the another said transaction;

if not, granting, via the transaction manager, a lock to ~~an accessor associated with the thread~~ and holding the lock until an owner of the thread delists the resource object, wherein the wrapper object is used to access the resource object for the thread.

32. (Previously Presented) The system of Claim 12, wherein:

the wrapper object is periodically garbage collected to clear state and unused locks.

33. (Previously Presented) The method of Claim 22, further comprising:

determining, via the transaction manager, whether an application associated with the thread is a specific type of application;

granting, via the transaction manager, the thread a lock only when the application is determined to be the specific type of application.

34. (Currently Amended) A system for ~~interleaving~~ supporting resource enlistment synchronization, comprising:

an application server with a plurality of threads, wherein the application server runs running on one or more processors and is associated with a plurality of applications, wherein each application is associated with at least one said thread;

at least one resource object, wherein the at least one resource object is associated with a resource connection object;

a transaction manager that manages a plurality of transactions, wherein each transaction is associated with a said application ~~at least one said thread, wherein the transaction manager maintains an enlistment data structure to manage resource object enlistment for the plurality of transactions, and wherein the enlistment data structure maintains a mapping between the one or more resource objects and the plurality of transactions;~~

wherein the resource connection object operates to perform

receiving a request, to access the at least one resource object that is associated with the resource connection object, from a [[first]] application_of the plurality of applications that runs on [[the]] a [[first]] thread_of the plurality of threads,
placing a call to the transaction manager and informing the transaction manager that current work performed by the at least one resource object is to be associated with a current transaction that is associated with the application,
wherein, after ~~the transaction manager receives~~ receiving the call from the resource connection object, the transaction manager operates to perform
first ~~checks~~ checking to see if there is an in-progress enlistment of the at least one resource object by another thread in another transaction,
if there is a lock, ~~blocks~~
blocking the request to enlist the resource object in the transaction and preventing different transactions ~~enlisted with~~ from enlisting a logical connection to the at least one resource object at same time, and
initiating the at least one resource object to perform work associated with the thread and the current transaction, after the at least one resource object is delisted from another transaction that owns the lock,
if there is no lock, ~~enlists~~
enlisting the at least one resource object in the transaction and signals signaling the at least one resource object to begin processing the request,
receiving from the at least one resource object a delist resource method call, after a result is obtained for the request, to delist the at least one resource object from the current transaction and provides the result to the first said application, and
initiating the at least one resource object to perform work associated with another thread and another transaction.

35. (New) The system of Claim 12, wherein:

the transaction manager maintains an enlistment data structure to manage resource object enlistment for the plurality of transactions, and wherein the enlistment data structure maintains a mapping between the plurality of resource objects and the plurality of transactions.